Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт математики и фундаментальной информатики
Базовая кафедра вычислительных и информационных технологий

УТВЕРЖДАЮ
Заведующий кафедрой
_____ / В.В. Шайдуров

«16» июня 2017 г.

БАКАЛАВРСКАЯ РАБОТА

Направление 02.03.01 Математика и компьютерные науки

ПРИМЕНЕНИЕ МЕТОДОВ КЛАСТЕРИЗАЦИИ ДЛЯ АНАЛИЗА ИСПОЛЬЗОВАНИЯ ИНТЕРНЕТ-РЕСУРСОВ

Научный руководитель
кандидат технически наук,
доцент _____ / С.В. Исаев
16.06.17

Выпускник _____ / Д.Ю. Донцов
16.06.17

Красноярск 2017

РЕФЕРАТ

Выпускная квалификационная работа по теме «Применение методов кластеризации для анализа использования интернет-ресурсов» содержит 39 страниц текста, 15 рисунков, 4 таблицы и 17 использованных источников.

КЛАСТЕРИЗАЦИЯ, КЛАСТЕРНЫЙ АНАЛИЗ, ОБРАБОТКА ЛОГ-ФАЙЛОВ, ОБРАБОТКА ДАННЫХ, МЕТОД ГЛАВНЫХ КОМПОНЕНТ, ЛИНЕЙНЫЙ ДИСКРИМИНАНТНЫЙ АНАЛИЗ.

Цель работы – исследовать применение методов кластеризации для анализа использования интернет ресурсов.

В процессе работы был произведен кластерный анализ пользователей по данным, полученным с прокси-сервера и были сформированы устойчивые группы пользователей, схожих по предпочтениям.

В результате исследований было выявлено, что метод иерархической кластеризации хорошо применим для анализа использования интернет-ресурсов, полученных с прокси-сервера.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Интернет постоянно растет и развивается, и каждый его пользователь оставляет свой след, используя его. Большая часть людей считают, что, просто пользуясь интернетом, ничего не загружая и не отправляя, они не оставляют о никакой информации себе во всемирной веб-паутине. Но это не так, ведь простой интернет-серфинг предоставляет о пользователе большое количество информации, даже не учитывая того, что он выкладывает в социальные сети. Под этой информацией подразумеваются истории поисковых запросов и посещенных сайтов, время серфинга, тип устройства, местоположение, браузер, ip-адрес и многое другое. Методы обработки и анализа подобного рода информации в огромных количествах называются одним общим термином Big-data.

Big-data работает с очень разнообразными данными, большая часть из которых не имеют фиксированной структуры, поэтому задача анализа этой информации является довольно сложной и интересной. В результате анализа нужной выборки данных можно получить детальные сведения о объекте или группе объектов.

В современном мире «Большие данные» широко используется в IT, Web, моделировании, бизнесе и прочих областях человеческой деятельности. Типичный пример Big-data — это сведения, поступающие с различных физических экспериментальных установок — например, с Большого адронного коллайдера, постоянно производящего колоссальное количество данных. Установка непрерывно выдает огромные объемы данных, с помощью которых ученые пытаются решать множество различных задач [6]. Еще одним примером использования Big-data являются все поисковые системы, основанные на обучающихся по этим «данным» алгоритмам. Они запоминают историю поисковых запросов пользователя, страницы, которые он посещает, его предпочтения, и с помощью полученной информации могут подсказывать

пользователю интересующие его поисковые запросы, или дополнять их, а также сначала выдавать результаты, которые считают наиболее полезными. Но на этом использование результатов анализа интернет активности не заканчивается. Уже частично тестируется поиск преступников, предотвращение терактов, а также определение суицидальных наклонностей только по истории интернет-сёрфинга.

В сфере бизнеса «Большие данные» не менее актуальны, чем в Web и IT. Имея необходимые данные, например, предпочтения покупателя, магазины могут предлагать одежду, которая придется ему по вкусу или книги, сюжет которых его явно заинтересует.

Все эти, а также другие задачи можно решать различными алгоритмами, одним из которых является «Кластерный анализ», служащий для разбиения множества объектов определенной структуры на подмножества по некоторым комбинированным признакам. Главной его особенностью является отсутствие фиксированного набора параметров для разбиения. Разбиение происходит по совокупности признаков, таким образом, что объекты одного множества имеют примерно одинаковые характеристики.

Суммируя сказанное, можно заключить, что кластерный анализ больших данных является актуальной задачей на сегодняшний день.

В процессе работы прокси-сервера, обеспечивающего пользователям доступ в интернет, ведутся записи информации об интернет активности каждого пользователя, в целях безопасности и мониторинга системы.

Прокси-сервер – это сервер, выступающий промежуточным слоем или посредником между клиентами и ресурсами, находящимися на другом сервере. В качестве ресурса может выступать любая доступная информация из мировой сети, так как вся она хранится на различных серверах.

На основе данных об интернет активности можно решать различные задачи: оптимизировать систему, уменьшая нагрузку путем распределения

ресурсов, улучшить ее защиту, отслеживать действия, которые могут навредить системе, или пользователей с подозрительной активностью.

Каждая запись в журнале проси-сервера содержит достаточное количество информации о пользователе и запросе для последующего анализа, а именно:

1) Данные о пользователе: логин и ip-адрес.
2) Данные о запросе: время выполнения, размер, метод, URL-адрес и тип контента.

Анализируя полученный набор данных, можно разделить множество пользователей прокси-сервера на некоторые подгруппы, например, по предпочитаемым тематикам сайтов, и выделить группы пользователей, которые держатся вместе на протяжение длительного периода и, следовательно, имеют схожие предпочтения.

1 Кластеризация

1.1 Что такое кластеризация

«Кластеризация — это автоматическое разбиение элементов некоторого множества на группы в зависимости от их схожести. Элементами множества могут быть что любые объекты, например, данные или вектора характеристик. Сами же группы принято также называть кластерами» [4].

Кластеризацию часто сравнивают со схожей процедурой – классификацией. Их отличие заключается в том, что в классификации множество результирующих групп задано четко, а в кластеризации оно определяется самим алгоритмом в процессе его работы.

Существует множество практических применений кластеризации как в информатике, так и в других областях. Вот несколько примеров применения кластеризации [7]:

1) Анализ данных
    - Упрощение работы с информацией
    - Визуализация данных
2) Извлечение и поиск информации
    - Построение удобных классификаторов
3) Группировка и распознавание объектов
    - Распознавание образов
    - Группировка объектов

Кластерный анализ можно представить в виде следующей последовательности действий [3]:

1) Выбор множества объектов.
2) Определение множества переменных, для оценки объектов и составление векторов характеристик.
3) Нормирование векторов характеристик одним из доступных методов.

4) Определение сходства между объектами по заданной метрике.
5) Применение выбранного метода кластерного анализа для разбиения множества объектов на кластеры по из степени схожести.
6) Представление результатов анализа.

Проанализировав результаты кластеризации можно скорректировать выбранные параметры, метрику или метод кластеризации, для улучшения результатов. Проводя данные улучшения можно прийти к наилучшему результату.

1.2 Формальные определения

Для дальнейших рассуждений, введем понятия, которыми будем оперировать.

Объектом будем называть элементарный набор данных, с которым работает алгоритм кластеризации.

Для каждого объекта определяются параметры, описывающие его, которые объединяются в *вектор характеристик* $x = (x_1, x_2, ..., x_m)$, где m – размерность пространства характеристик, а x_i – отдельная *характеристика объекта* (качественная или количественная).

Меру сходства двух объектов $d(u, v)$ вычисленную по заданной метрике будем называть *расстоянием* между объектами, где u, v – элементы множества.

1.3 Выделение вектора характеристик

Первым шагом необходимо выделить характеристики объектов, которые будут использованы в процессе кластеризации. Это могут быть как количественные (рост, вес, координаты, счетчики, …) так и качественные характеристики (цвет, статус, настроение, …).

Чаще всего работают с количественными характеристиками, так как для них применимо большое число метрик.

На большом пространстве характеристик, процесс кластеризации происходит довольно медленно, и его результаты не всегда приемлемы. Поэтому, при большой размерности пространства характеристик, нужно постараться его уменьшить, оставив наиболее важные свойства объектов.

Получившийся набор характеристик каждого объекта необходимо нормализовать, для лучших результатов. *Нормализовать вектор*, значит привести его к фиксированному размеру. Характеристики каждого нормализованного вектора будут лежать в фиксированном отрезке, например, $[0;1]$или $[-1;1]$, в зависимости от поставленной задачи.

1.4 Меры расстояний

После выявления вектора характеристик необходимо выбрать функцию для определения степени сходства двух объекта, называемую мерой расстояний. Выбранная функция должна удовлетворять всем условиям метрики [2, стр. 5].

Существуют различные метрики для вычисления близости объектов. Обозначив $u,v-$ объекты между которыми вычисляется расстояние $d(u,v)$ и u_i,v_i – их координаты, опишем некоторые из существующих метрик:

1) **Евклидово расстояние**. Классическая метрика Евклида, являющаяся геометрическим расстоянием в многомерном пространстве

$$d(u,v)=\sqrt{\sum_{i=1}^{m}(u_i-v_i)^2}.$$

2) **Квадрат евклидова расстояния**, равный евклидову расстоянию, возведенному в квадрат. Используется для увеличения веса более отдаленных друг от друга объектов.
3) **Расстояние городских кварталов или манхэттенское расстояние.** Вычисляется как средняя разность по координатам и чаще всего приводит к результатам аналогичным обычному расстоянию Евклида.

$$d(u,v) = \sum_{i=1}^{m} |u_i - v_i|.$$

4) **Степенное расстояние**, применяемое при необходимости изменить вес, в большую или меньшую сторону, относящийся к размерности, для которой соответствующие объекты сильно отличаются. Степенное расстояние вычисляется по формуле, схожей с формулой расстояния Евклида:

$$d(u,v) = \sqrt[r]{\sum_{i=1}^{m} (u_i - v_i)^p},$$

где r и p – параметры, определяемые пользователем. Параметр p отвечает за постепенное взвешивание разностей по отдельным координатам, а параметр r ответственен за прогрессивное взвешивание больших расстояний между объектами. Как было сказано раннее, при значениях обоих равны двум, данная метрика совпадает с расстоянием Евклида.

5) **Расстояние Чебышева.** Мера, применяемая ввиду необходимости определить два объекта как различные, при какой-то одной отличной координате. Расстояние Чебышева вычисляется по формуле

$$d(u,v) = \max(|u_i - v_i|).$$

От выбора метрики во многом зависят результаты кластеризации, и для различных метрик они могут существенно отличаться.

1.5 Классификация алгоритмов

1.5.1 Иерархические и плоские

Плоские алгоритмы разбивают заданное множество объектов на кластеры, строя единственное разбиение, и для получения другого разбиения, необходимо повторять процесс кластеризации с другими параметрами.

Иерархические алгоритмы, в отличии от плоских, создают не единственное разбиение, а систему вложенных разбиений на непересекающиеся кластеры. В результате выполнения этого алгоритма получается дерево разбиений, корнем которого является кластер, содержащий все множество объектов, а листьями — более мелкие кластера.

1.5.2 Четкие и нечеткие

Данная классификация определяет, может ли один объект выборки принадлежать одновременно нескольким кластерам, или он всегда принадлежит единственному кластеру.

В четких (или непересекающихся) алгоритмах каждый объект выборки принадлежит только одному кластеру, т.е. каждому объекту сопоставляется единственный номер кластера, которому он принадлежит. В нечетких (или пересекающиеся) алгоритмах каждому объекту сопоставляется набор вещественных значений, отображающих вероятность отношения данного объекта к каждому из кластеров. Другими словами, в нечетких алгоритмах, каждый объект принадлежит всем кластерам с разной степенью [1].

1.6 Объединение кластеров

Как было сказано раннее, некоторые алгоритмы кластеризации выполняют разбиение ступенчато, а именно на каждом шаге два наиболее близко расположенных объекта объединяются и рассматриваются как один кластер. В связи с этим возникает необходимость введения меры расстояний между кластерами, для определения их близости [10].

Вот несколько способов, вычисления расстояния между кластерами:

1) **Одиночная связь (Ближайший сосед).**

   В данном способе, расстоянием между кластерами считается расстояние между двумя наиболее близкими объектами (ближайшими соседями) в различных кластерах. Т.е. вычисляется расстояния между всеми возможными парами объектов из различных кластеров и после этого вычисляется минимальное из этих расстояний. Этот «минимум» и считается расстоянием между кластерами.

2) **Полная связь (Наиболее удаленный сосед).**

   Расстояния между кластерами вычисляется аналогично Одиночной связи, но вместо минимального расстояния вычисляется максимальное, т.е. расстояние между наиболее удаленными соседями. Данный способ, как правило, работает довольно хорошо, если кластеры имеют форму близкую к сферической. Если же кластеры являются «цепочечными» или имеют удлиненную форму, то этот метод непригоден.

3) **Невзвешенное попарное среднее.**

   Расстояние между двумя отличными друг от друга кластерами можно определить, как среднее расстояние между всеми парами объектов из различных кластеров. Данный метод довольно эффективен, при условии, что объекты формируют различные группы, однако он работает также хорошо и в случаях, протяженных или «цепочечного» типа кластеров.

4) **Взвешенное попарное среднее.**

Этот метод схож с методом невзвешенного попарного среднего, однако в нем, при вычислении расстояний учитывается размер соответствующих кластеров, (т.е. число объектов, содержащихся в них), и используется в качестве весового коэффициента. В связи с этим данный метод необходимо использовать, если ожидаются кластеры, значительно отличающиеся по размерам.

5) **Невзвешенный центроидный метод.**

Для вычисления расстояния этим методом, необходимо вычислить центры тяжести каждого из кластеров, а после этого считать за расстояние между кластерами расстояние между их центрами тяжести. При вычислении центра тяжести вес каждого объекта считается равным единице.

6) **Взвешенный центроидный метод (медиана).**

Этот метод идентичен предыдущему, за исключением того, что при вычислениях используются веса для учета разницы между размерами кластеров. Поэтому, если имеются или подозреваются значительные отличия в размерах кластеров, этот метод оказывается предпочтительнее предыдущего.

1.7 Алгоритмы кластеризации

1.7.1 Алгоритмы иерархической кластеризации

Алгоритмы иерархической кластеризации принято разделять на два типа: нисходящие и восходящие алгоритмы. Нисходящие, действуют по принципу «от большего к меньшему»: в начале процесса все объекты помещаются в единственный кластер, вершину дерева, после чего, на каждом шаге, один из существующих кластеров разбивается на два более мелких, пока каждый объект не будет принадлежать собственному кластеру.

Второй тип алгоритмов, восходящие, более распространен, и работает в обратную сторону, относительно первого. Сначала каждый из объектов

помещается в собственный кластер, и на каждом шаге алгоритма, два ближайших кластера объединяются в один, до тех пор, пока не останется единственный кластер, содержащий всю выборку объектов [5, стр. 40].

Результатом кластеризации данным алгоритмом является дерево разбиений, называемое дендрограммой. Наиболее популярный пример использования иерархической кластеризации - классификация животных и растений.

1.7.2 Алгоритмы квадратичной ошибки

Задачу кластеризации можно интерпретировать иначе: необходимо построить оптимальное разбиение объектов на группы. При этом условие оптимальности может быть задано требованием минимизации среднеквадратической ошибки разбиения:

$$e^2(X) = \sum_{j=1}^{k} \sum_{i=1}^{n_j} \left\| x_i^{(j)} - c_j \right\|^2,$$

где X – множество объектов x^j, x_i^j – их координаты, c_j — «центр масс» кластера j (считая массу каждой точки равной единице).

Алгоритмы данной категории относятся к классу плоских алгоритмов. Метод k-средних считается наиболее популярным в этой категории, ввиду того, что алгоритм разбивает заданное множество объектов на указанное число кластеров, расположенных на как можно большем расстоянии друг от друга. Работа этого метода разбивается на несколько этапов:

1) Случайно выбрать k начальных «центров масс» кластеров.
2) Отнести каждый объект к кластеру с ближайшим «центром масс».
3) Пересчитать «центры масс» кластеров согласно их текущему составу.
4) Проверить критерий остановки, и в случае его не выполнения, вернуться к пункту 2.

В качестве критерия остановки работы алгоритма как правило используют минимальное изменение среднеквадратической ошибки. Также работа алгоритма завершается, если на шаге 2 не было объектов, сменивших свой кластер.

Результат использования данного алгоритма на плоскости, близок к диаграмме Вороного. К недостаткам этого алгоритма можно отнести необходимость задавать число кластеров для разбиения.

1.7.3 Нечеткие алгоритмы

Как было сказано выше, алгоритмы нечеткой кластеризации относят каждый объект к каждому кластеру с некоторой вероятностью, в отличии от четких методов. Алгоритмов данной категории не слишком много, ввиду этого рассмотрим наиболее популярный – метод c-средних (c-means). Этапы работы алгоритма схожи с этапами метода k-средних:

1) Задать начальное нечеткое разбиение n объектов на k кластеров путем выбора матрицы принадлежности U размера $n*k$.
2) Найти значение критерия нечеткой ошибки, используя матрицу U

$$E^2(X,U)=\sum_{i=1}^{N}\sum_{j=1}^{K}U_{ij}\left\|x_i^{\,j}-c_j\right\|^2\ ,\ c_k=\sum_{i=1}^{N}U_k x_i\ ,$$

где X – множество объектов x^j, $x_i^{\,j}$ – их координаты, c_j — «центр масс» кластера j (считая массу каждой точки равной единице), U_{ij} - матрица принадлежности.

3) Перегруппировать объекты с целью уменьшения этого значения критерия нечеткой ошибки.
4) Возвращаться в п. 2 до тех пор, пока изменения матрицы U не станут незначительными.

Этот алгоритм стоит применять только если заранее известно число кластеров и необходимо вычислить отношение каждого объекта ко всем кластерам.

1.7.4 Алгоритмы, основанные на теории графов

Особенность графовых алгоритмов в том, что вся выборка объектов представляется в виде графа $G = (V, E)$, где V – множество вершин и E – множество ребер, в роли вершин которого выступают объекты выборки, а вес ребер равен расстоянию между объектами, которые они соединяют. Преимуществами алгоритмов данной категории являются их наглядность и относительная простота реализации с возможностью внесения модернизаций, основанных на геометрических суждениях. В этой категории наиболее популярными являются алгоритм выделения связных компонент, алгоритм построения минимального покрывающего (остовного) дерева и алгоритм послойной кластеризации.

1.7.4.1 Алгоритм выделения связных компонент

Для работы данного алгоритма необходим параметр R, задающий пороговое значение для весов ребер. В процессе работы этого алгоритма постепенно удаляются все ребра, вес которых превышает пороговое значение. В результате работы получается граф, в котором остаются только ребра, соединяющие наиболее близкие объекты.

Чтобы получить кластеры, остается только подобрать значение R так, чтобы граф разделился на несколько связных компонент, которые и будут являться кластерами.

Чаще всего, для подбора параметра R пользуются построением гистограммы распределений попарных расстояний. Если кластерная структура выражена относительно хорошо, то гистограмма будет иметь два пика, один из

которых соответствует внутрикластерным расстояниям, а второй – межкластерным. Параметр *R* подбирается из зоны минимума между этими пиками.

Минус этого алгоритма в довольно сложном управлении результирующим количеством кластеров.

1.7.4.2 Алгоритм минимального покрывающего дерева

Суть этого алгоритма в построении минимального покрывающего дерево, и последовательном удалении ребер с наибольшим весом. На рис. 1 изображено минимальное покрывающее дерево, полученное для десяти объектов, являющихся точками на плоскости.

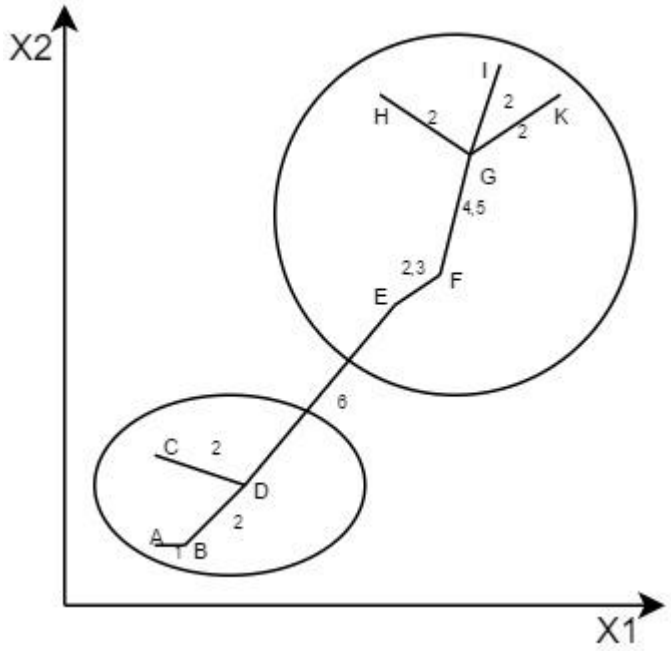

Рисунок 1 - Минимальное покрывающее дерево

Путём удаления связи с максимальным весом, помеченной DE, получаем две компоненты: $\{A,B,C,D\}$ и $\{E,F,G,H,I,K\}$. Второй кластер в дальнейшем может быть разделён ещё на два кластера путём удаления максимального ребра FG, которое имеет вес, равный 4,5 единицам.

1.7.4.3 Послойная кластеризация

В основе послойной кластеризации лежит на выделение связных компонент графа на заданном уровне расстояний между вершинами, как и в алгоритме выделения связных компонент, рассмотренном ранее. Уровень расстояния задается порогом расстояния *c,* лежащим в диапазоне всех «расстояний» в графе $c \in [\rho_{\min}, \rho_{\max}]$, где $\rho_{\min}$ и $\rho_{\max}$ - минимальное и максимальное расстояния в кластере соответственно.

Изменяя пороговое значение параметра *c*, можно получить последовательность подграфов исходного графа G, которые отображают иерархическую связь между кластерами:

$$G^0 \subseteq G^1 \subseteq ... \subseteq G^m ,$$

где $G_t = (V, E_t)$ - граф на уровне c_t, $E_t = \{e_{ij} \in E, p_{ij} \le c_t\}$ - множество ребер, c_t - порог расстояний на уровне t, m – количество уровней иерархии, G_0 - пустое множество, $G_m = G$, то есть все множество объектов.

Посредством изменения порогов расстояния $\rho_{\min} = c_0 < c_1 < ... < c_m = \rho_{\max}$, возможно контролировать глубину иерархии получаемых кластеров. Таким образом, алгоритм послойной кластеризации способен создавать как плоское разбиение данных, так и иерархическое.

1.8 Сравнение алгоритмов

Одной из главных характеристик любого алгоритма является вычислительная сложность. В таблице 1 приведены вычислительные сложности каждого из алгоритмов.

Таблица 1- Вычислительная сложность алгоритмов кластеризации

Алгоритм	Вычислительная сложность
Иерархический	$O(n^2)$
K-средних	$O(nkl)$, где k и l число кластеров и итераций соответственно
C-средних	
Выделение связных компонент	Зависит от выбранного алгоритма
Минимальное покрывающее дерево	$O(n^2 \log(n))$
Послойная кластеризация	$O(max(n,m))$, где $m < n(n-1)/2$

Кроме вычислительной сложности, каждый алгоритм отличается набором входных данных, результатов, а также формой получившихся кластеров.

Таблица 2- Формат данных алгоритмов кластеризации

Алгоритм	Форма кластеров	Входные данные	Результат
Иерархический	Произвольная	Число кластеров и порог расстояния	Бинарное дерево кластеров
K-средних	Гиперсфера	Число кластеров	Центры кластеров
C-средних	Гиперсфера	Число кластеров, степень нечеткости	Центры кластеров, матрица принадлежности
Выделение связных компонент	Произвольная	Порог расстояния R	Древовидная структура кластеров
Минимальное покрывающее дерево	Произвольная	Число кластеров или порог расстояния	Древовидная структура кластеров
Послойная кластеризация	Произвольная	Последовательность порогов расстояния	Древовидная структура кластеров с разными уровнями иерархии

1.9 Библиотеки с реализованной кластеризацией

На данный момент все методы кластеризации уже реализованы в отдельных библиотеках или функциях на многих языках программирования. Это позволяет быстро внедрять кластерный анализ в свое решение, с наименьшими трудностями.

Вот список некоторых библиотек:

1) Matlab - Statistics Toolbox, Fuzzy Logic Toolbox

2) Python [12] - Scipy.cluster [8], Pycluster
3) Ruby – kmeans-clusterer (OpenSource проект)
4) C/C++ - The Open Source C Clustering Library

Выбор языка программирования для решения задачи кластеризации лежит полностью на программисте. Если же в выбранном языке нет готовых решений кластеризации, то их написание не составляет труда, благодаря большому количеству информации в учебниках и сети интернет.

2 Проектное решение и архитектура системы

Для решения поставленной задачи был выбран язык программирования Python версии 3.5, и все библиотеки, используемые дальше, используются для выбранного языка.

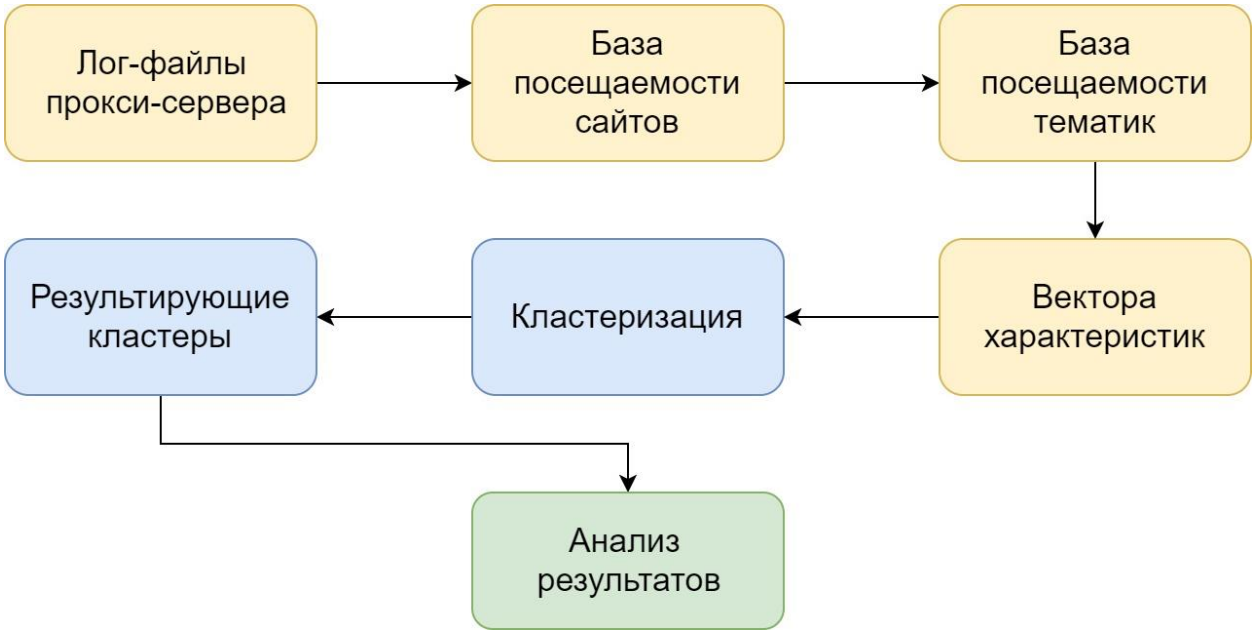

Рисунок 2 - Алгоритм решения поставленной задачи

2.1 Входные данные

Имеются лог-файлы прокси-сервера за каждый день его функционирования в течение месяца. Данные за различные периоды(недели) разделены на разные каталоги, причем каждый содержащий информацию за один день представлен в формате gz-архив. Данный формат архивов имеет высокую степень сжатия именно для лог-файлов и довольно прост в распаковке [9].

В каждой строке лог-файла содержится различная информация о запросе, созданным любым пользователем прокси сервера, такая как [16, 17]:

1) Тип запроса (подключения)
2) Время запроса

3) URL-адрес
4) Тип контента
5) Имя пользователя, выполнившего запрос
6) Код результата запроса
7) и т.д.

2.2 Предобработка

Во входных данных содержится много лишней информации, которая не будет использоваться в дальнейшем, поэтому из каждой записи важно выделить только необходимую информацию, и сохранить в более компактном виде.

Первым шагом, из всей информации, представленной в каждой записи, было выделено три основных параметра:

1) URL
2) Имя пользователя (UserName)
3) Код результата запроса

Вторым шагом было отбрасывание всех строк с кодом результата запроса отличным от 200, так как, «200 OK» - это код успешно выполненного запроса [17].

Каждый URL содержит в себе не только адрес сайта, но и путь к необходимой странице сайта с параметрами GET-запроса. Для последующей работы, из строки URL требуется извлечь и сохранить только домен сайта.

После этих простых манипуляций, полученные данные UserName:Domain были занесены в словарь, ключом которого служит имя пользователя(UserName), а значение – еще один словарь, в котором ключом является домен сайта, посещенного пользователем, а значением - счетчик посещений этого домена выбранным пользователем (рис. 3).

Рисунок 3 - Формат базы посещаемости доменов

Использование данной структуры хранения данных позволило компактно хранить только полезные данные извлеченные из лог-файлов. Используя язык программирования Python, после обработки лог-файлов за одну неделю, в каждом из которых в среднем 2-3 миллиона записей, вместо исходных 300Мб архивированных данных (или 1.6ГБ распакованных данных) был получен файл размером всего 2-3Мб, в котором хранится упакованная база.

2.3 Получение тематик сайтов

Данные, хранящиеся в полученной базе можно представить в виде многомерного пространства, в котором каждая координата — это число посещений конкретного домена конкретным пользователем.

Ввиду того, что различных доменов получилось довольно много, и даже после отсечения «случайно-посещенных» сайтов их осталось около 4000, то разумно предположить, что кластеризация пространства такой размерности не даст никакого наглядного результата, и процесс ее выполнения будет достаточно длительным. В связи с этим, необходимо произвести некоторые манипуляции для сужения пространства до меньшей размерности.

Для решения задачи сужения размерности пространства было принято решение перейти от подсчета посещений по доменам сайтов к подсчету по тематикам сайтов. Это позволит получать предпочтения пользователей одновременно с процессом кластеризации. Встает вопрос об определении тематики заданного сайта.

Чтобы решить проблему определения тематики сайта, можно воспользоваться сервисом Яндекс.Каталог (https://yandex.ru/yaca). Данный сервис предоставляет различную информацию об указанном сайте, одной из которых является «Рубрика», чем можно и воспользоваться. К сожалению, тематики некоторых не популярных или малопосещаемых сайтов не определяются Яндексом, поэтому такие сайты придется отбросить.

Определив тематику для каждого сайта, можно составить аналогичную предыдущему пункту базу (рис. 4).

Рисунок 4 - Формат базы посещаемости тематик

В результате, получившаяся база имеет размерность 160, которая в 250 раз меньше исходной размерности, что делает возможным выполнение кластерного анализа имеющихся данных.

Ввиду того, что в процессе кластеризации используются вектора характеристик, полученные данные следует перевести в матрицу и сохранить в csv-таблицу. Строками таблицы будут являться пользователи, столбцами – тематики, а каждая ячейка будет являться счетчиком посещений (табл. 3).

Таблица 3 – Посещаемость тематик

	Баннерные сети	Социальные сети	Интернет	Банки
Viktor	0	7	24	0
Arhimed	125	1	80	0
Denis	21	21	78	15
Masha	0	58	3102	0
Admin	0	2	142	0
Ivan	226	14	113	56
Sasha	73	8	227	8

Для сохранения и загрузки данных из csv-таблиц отлично подходит библиотека Pandas [13], используемая для обработки данных. С ее помощью можно удалять не нужные строки-столбцы сразу при загрузке матрицы. Это удобно использовать при выборке необходимых данных из таблицы.

2.4 Нормирование векторов

Как уже было сказано раннее, кластерный анализ лучше всего работает на множестве нормированных векторов, поэтому перед началом кластеризации необходимо произвести нормирование таблицы [9].

Встает выбор между формулами

$$X^* = \frac{X - X_{\min}}{X_{\max} - X_{\min}}, \; X^* = \frac{X}{X_{\max}},$$

где X^* - новое значение ячейки, X_{min} – минимальное значение в строке, X_{max} – максимальное значение в строке.

Нормирование векторов по первой формуле, укладывает все значения в отрезок $[0;1]$, причем значения 0 и 1 будут обязательно присутствовать в каждом векторе.

Использование второй формулы не гарантирует присутствие 0 в каждой строке. Характеристика будет принимать нулевое значение только при условии, что ее значение было равным нулю и до нормирования.

Исходя из поставленной задачи, правильнее будет использовать вторую формулу, так как очень важно даже минимальное значение характеристики, которое в первом случае отождествляется с нулем.

Данный процесс был реализован с помощью библиотеки NumPy [13], преимущество которой в том, что она позволяет работать со строками матрицы так же, как и с обычными ячейками. Например, процесс деления всей строки на максимальный элемент можно реализовать без циклов, в одну строку.

2.5 Кластеризация

Полученные нормированные данные можно кластеризовать. Для решения этой задачи хорошо подходит библиотека SciPy [8]. В этой библиотеке реализован модуль cluster.hierarchy, предоставляющим набор средств для иерархической кластеризации.

Для выполнения кластеризации использовалась соответствующая функция linkage(z, method, metric). Данная функция принимает следующие входные параметры [15]:

1) **Z** - массив векторов или матрица, описывающие объекты.
2) **Method** – метод вычисления расстояния между кластерами.
3) **Metric** – метрика для вычисления расстояния между точками. По умолчанию – Евклидова.

Функция linkage поддерживает множество методов вычисления расстояния между кластерами, при использовании которых получаются различные результаты. Обозначив U, V - кластера, $u[i], v[j]$ – их объекты соответственно, опишем некоторые из доступных методов [10, 11]:

Single. Расстояние между кластерами вычисляется как минимальное расстояние, среди расстояний между всеми парами точек из каждого кластера. Этот метод еще называется «Ближайший сосед»

$$d(U,V) = \min(d(u[i], v[j])),\ \forall i, j.$$

Complete. Метод аналогичен предыдущему, но вместо минимального расстояния, берется максимальное.

$$d(U,V) = \max(d(u[i], v[j])),\ \forall i, j.$$

Average. В данном методе расстояние вычисляется по формуле:

$$d(U,V) = \sum_{i,j} \frac{d(u[i], v[j])}{|u[i]| * |v[j]|},\ \forall i, j.$$

Weighted. В данном методе расстояние вычисляется по формуле:

$$d(U,V) = \frac{d(T,V) + d(S,V)}{2},$$

где кластер U был получен путем объединения кластеров T и S, а V – не использованный кластер.

Centroid. В данном методе расстояние между кластерами считается как расстояние между специальными точками – центроидами кластеров. Если два кластера объединяются в один, то центроид получившегося кластера высчитывается заново, на новом множестве точек. Центроид можно считать центром тяжести кластера. Расстояние между центроидами высчитывается по указанной метрике.

$$d(U,V) = \|c_U - c_V\|_2,$$

где c_U, c_V - центры кластеров U, V соответственно.

Ward. Данный метод использует алгоритм минимизации дисперсии Уорда. Расстояние вычисляется по формуле:

$$d(U,V) = \sqrt{\frac{|V|+|S|}{Q} * d(V,S) + \frac{|V|+|T|}{Q} * d(V,T) - \frac{|V|}{T} * d(S,T)^2},$$

где кластер U был получен путем объединения кластеров T и S, а V – не использованный кластер, а $Q = |V| + |T| + |S|$. Этот метод называется «Инкрементный алгоритм».

Экспериментальным путем было выявлено, что для поставленной задачи лучше всего подходит Инкрементный алгоритм, поэтому он был использован при кластеризации.

После выполнения, функция linkage возвращает массив связей, являющийся результатом иерархической кластеризации. Этот массив можно графически отобразить с помощью специальных функций или, выполнив функцию fcluster, получить разбиение на заданное число кластеров [14].

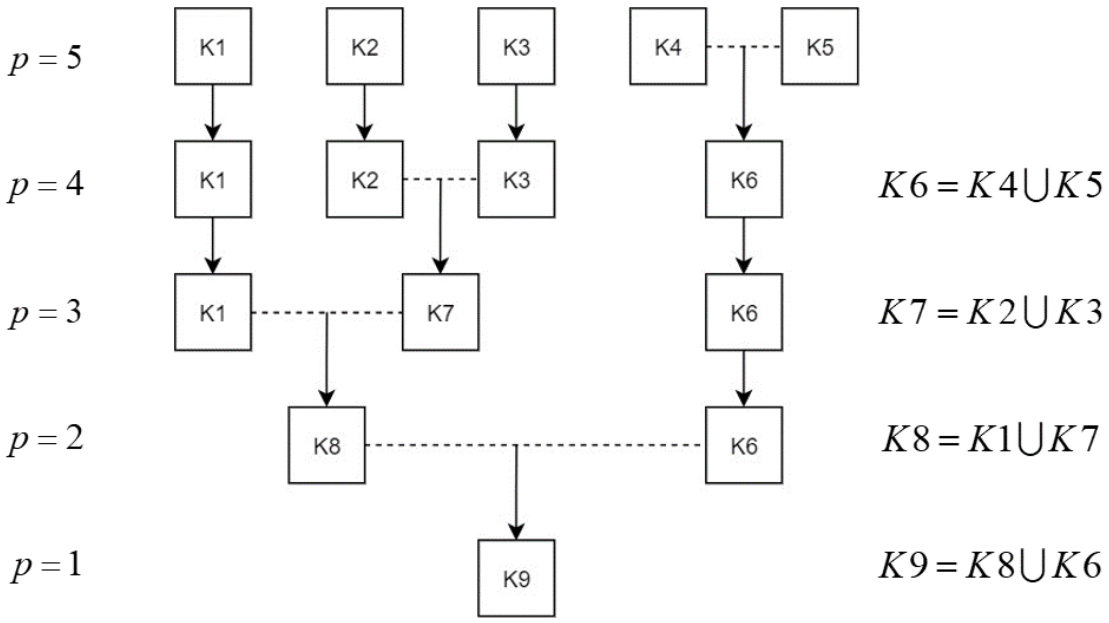

Рисунок 5 - Схема отображающая процесс кластеризации

3 Анализ результатов

3.1 Визуализация

Самым простым методом визуализации результатов иерархической кластеризации является **дендрограмма** – бинарное дерево разбиения кластеров. Выполнив функцию dendrogram из библиотеки SciPy была получена дендрограмма (рис. 6), отображающая процесс разбиения множества пользователей на кластеры. По оси абсцисс отображаются размеры кластеров, а по оси ординат – расстояние d между кластерами.

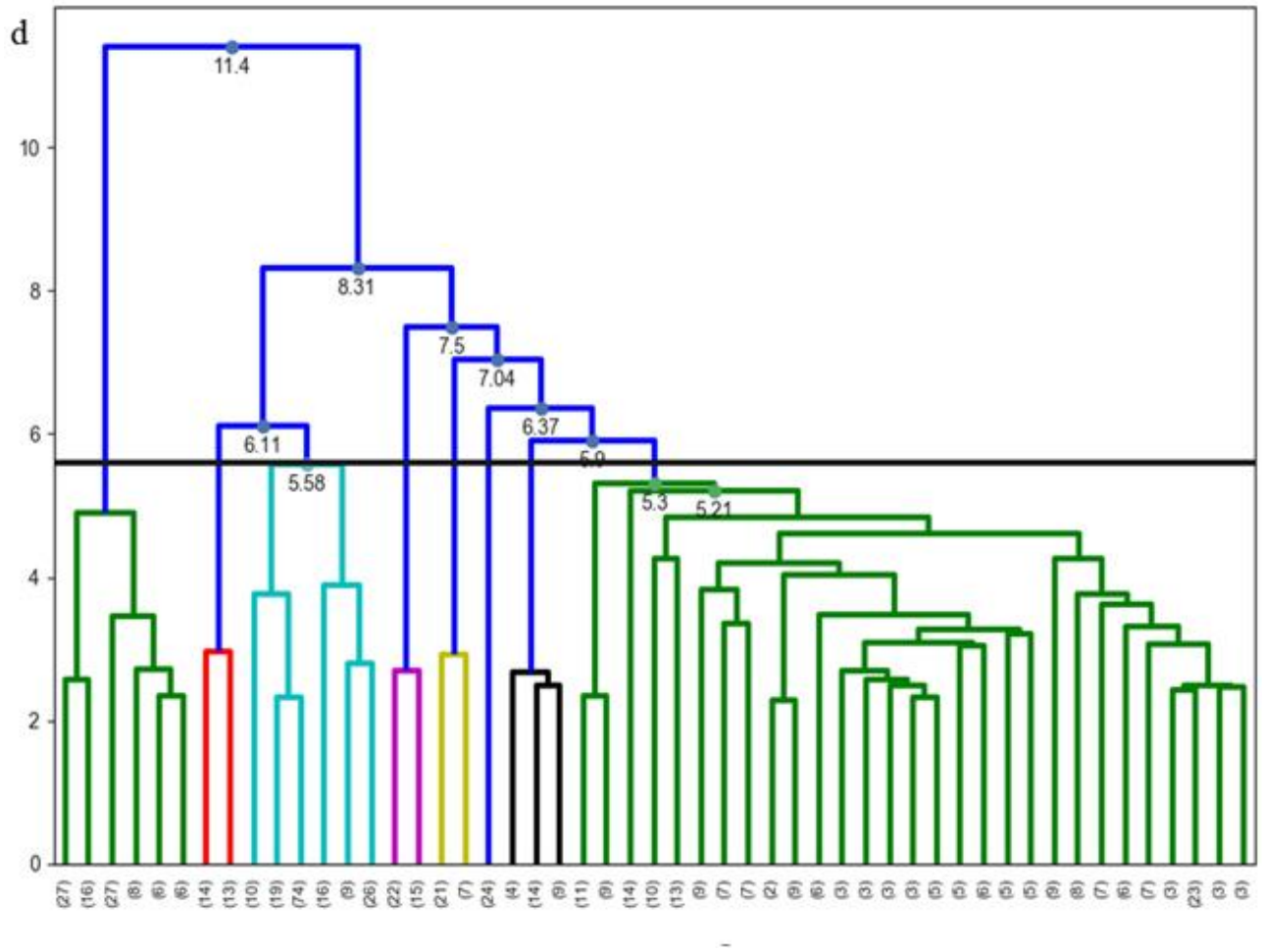

Рисунок 6 - Дендрограмма иерархической кластеризации

Данный метод визуализации показывает дерево разбиений, в данном случае - усеченное, но не дает достаточного представления об расположении объектов в

пространстве. Возникает проблема отображения многомерных кластеров на двухмерную плоскость. Поэтому, для проекции на двухмерную плоскость используются методы понижения размерности пространства.

Метод главных компонент (PCA) - один из основных способов уменьшить размерность данных, потеряв наименьшее количество информации. Другим языком, данный метод аппроксимирует n-мерное пространство объектов до n-мерного эллипсоида, полуоси которого, в дальнейшем, будут считаться главными компонентами. При проекции именно на эти оси сохраняется наибольшее количество информации.

На рис. 7 и 8 представлены результаты проекции данного множества объектов на главные компоненты при разном числе кластеров. Красной тенью на изображениях показана относительная плотность множества объектов.

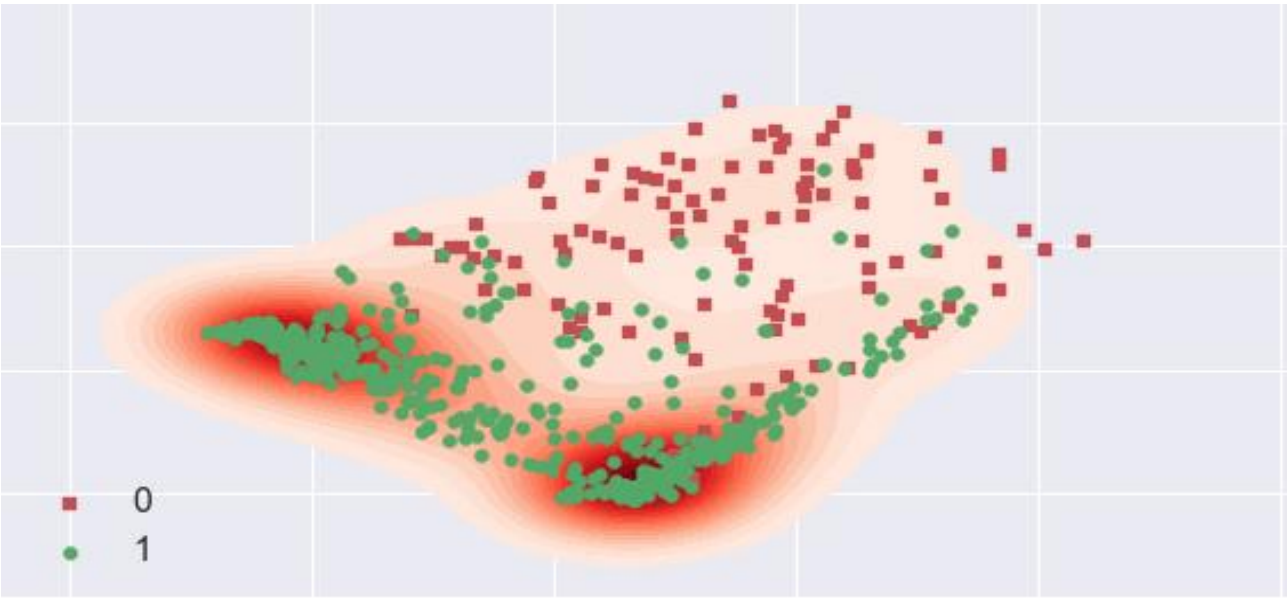

Рисунок 7- Результат применения метода главных компонент для 2 кластеров

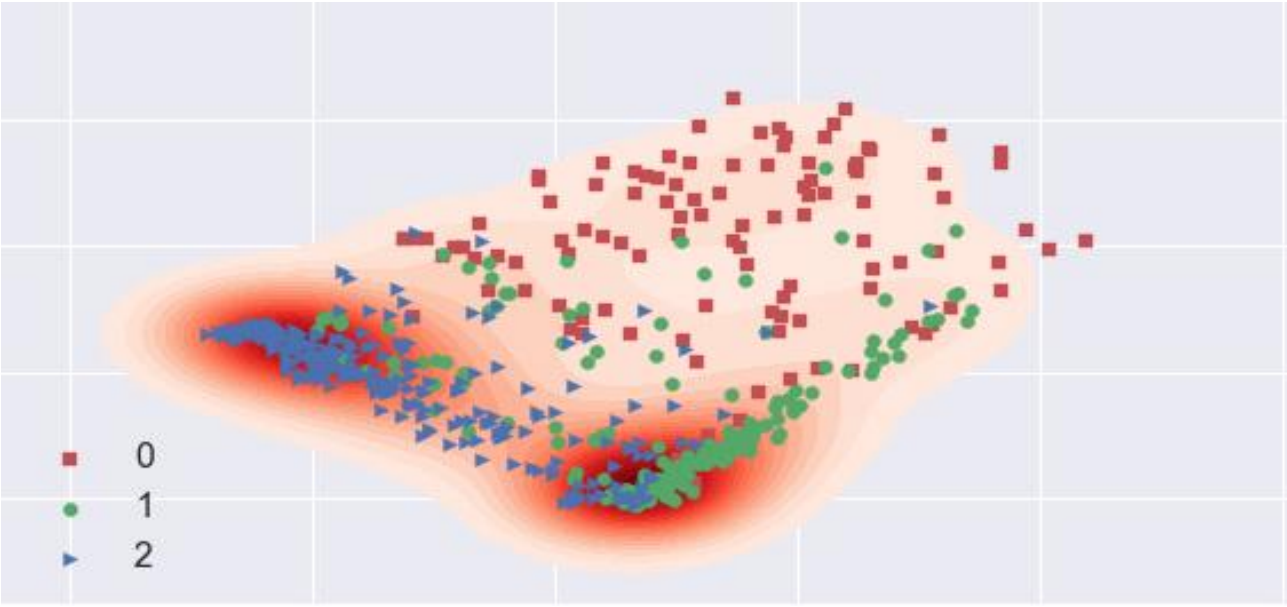

Рисунок 8 - Результат применения метода главных компонент для 3 кластеров

Как видно по рисунку, в результате применения метода главных компонент получились схожие изображения, отличающиеся только маркерами, отвечающими за принадлежность объекта к одному из кластеров. В этом минус использования данного метода для проекции различных разбиений, ведь его применение не показывает изменение расположения кластеров и их формы в зависимости от разбиения.

Для получения различных проекций, зависящих от разбиения, часто используют другой метод понижения размерности пространства - линейный дискриминантный анализ.

Линейный дискриминантный анализ (LDA) - это метод поиска линейной комбинации переменных, наилучшим образом разделяющей два или более класса. Чаще всего результаты LDA используют как часть линейного классификатора, однако другим возможным применением является понижение размерности входных данных.

В отличии от PCA, при использовании LDA, проекции при различном числе кластеров будут существенно отличаться, что поможет выявить оптимальное число кластеров, на которое необходимо разбивать данное

множество объектов. На рис. 9 и 10 отображены проекции кластеров пользователей с использованием линейного дискриминантного анализа в качестве метода понижения размерности пространства.

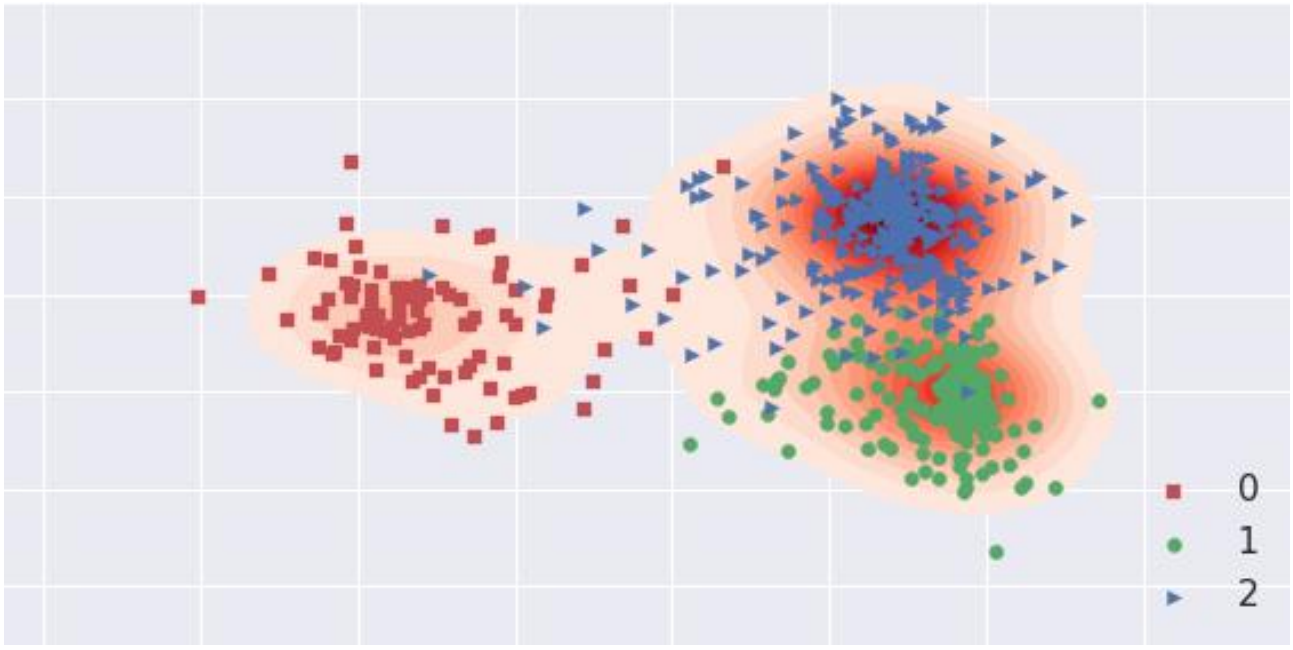

Рисунок 9 - Результат применения линейного дискриминантного анализа для 3 кластеров

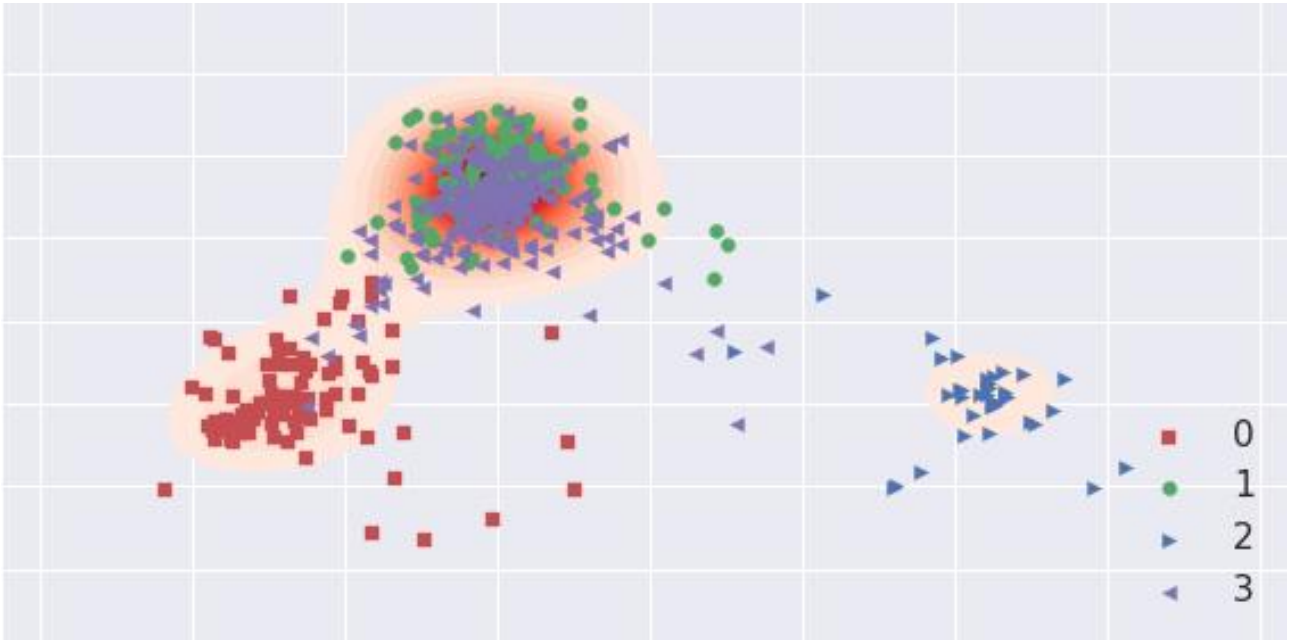

Рисунок 10 - Результат применения линейного дискриминантного анализа для 4 кластеров

Исследуя проекции можно заметить, что при числе кластеров равном 3 видно отчетливое разделение кластеров, в отличии от 4.

3.2 Оптимальное число кластеров

Одним из важных вопросов при решении задачи кластеризации является выбор необходимого числа кластеров. В некоторых случаях это число может быть задано априорно, однако в общем случае это число определяется в процессе разбиения множества на различное число кластеров [10].

Анализируя разбиения на различные числа кластеров, можно определить такое число кластеров, при котором кластеры разделены наилучшим образом. Такое число кластеров будем называть оптимальным для заданной выборки данных. Определить это число можно с помощью визуализации, рассмотренной выше, или другими различными методами.

Одним из таких методов является **метод локтя**. Этот метод заключается в построении функции изменения вариаций данных внутри кластеров при изменении числа кластеров. Число кластеров, при котором отмечено наибольшее изменение функции, можно считать оптимальным. Если таких чисел несколько, то все они могут являться оптимальными.

Как видно по графику, изображенному на рис. 11, при значениях количества кластеров равных 2 и 3 функция сильно изменяется, поэтому при дальнейшем анализе можно использовать только эти значения числа кластеров.

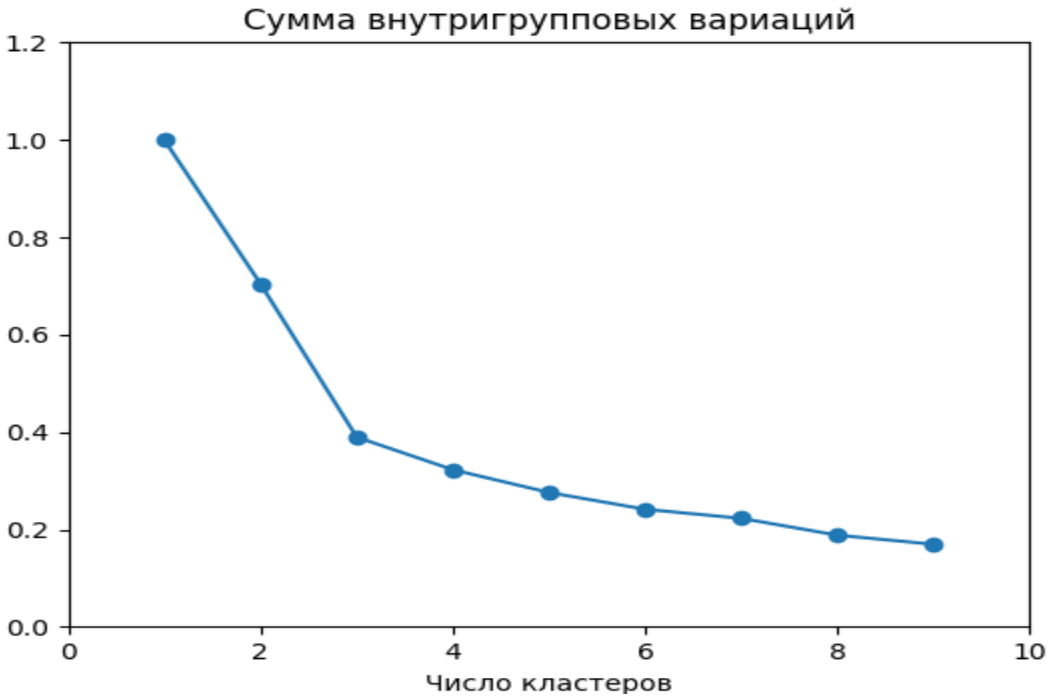

Рисунок 11 - График изменения суммы внутригрупповых вариаций от числа кластеров

3.3 Динамический анализ

Проведя кластерный анализ пользователей за один промежуток времени, например, за неделю, нельзя с уверенностью сказать, что пользователи, находящиеся в одном кластере, имеют действительно схожие предпочтения, а не оказались рядом случайно. Для проверки достоверности кластеризации можно выполнить аналогичные действия с данными взятыми из лог-файлов за другой, к примеру, последующий, промежуток времени, после чего сравнив результаты различных разбиений, можно отличить настоящих «соседей» по интересам, от пользователей, случайно попавших в кластер или перемещающихся между ними.

На рис. 12-13 представлены результаты разбиений пользователей по данным взятым за последовательные 4 недели на 5 и 8 кластеров соответственно. По оси абсцисс указаны промежутки времени, высота каждого столбика в одной группе равна числу пользователей в соответствующем кластере, а сумма всех пользователей равна 591. Основываясь на этих данных,

можно предположить, что устойчивые группы пользователей действительно присутствуют.

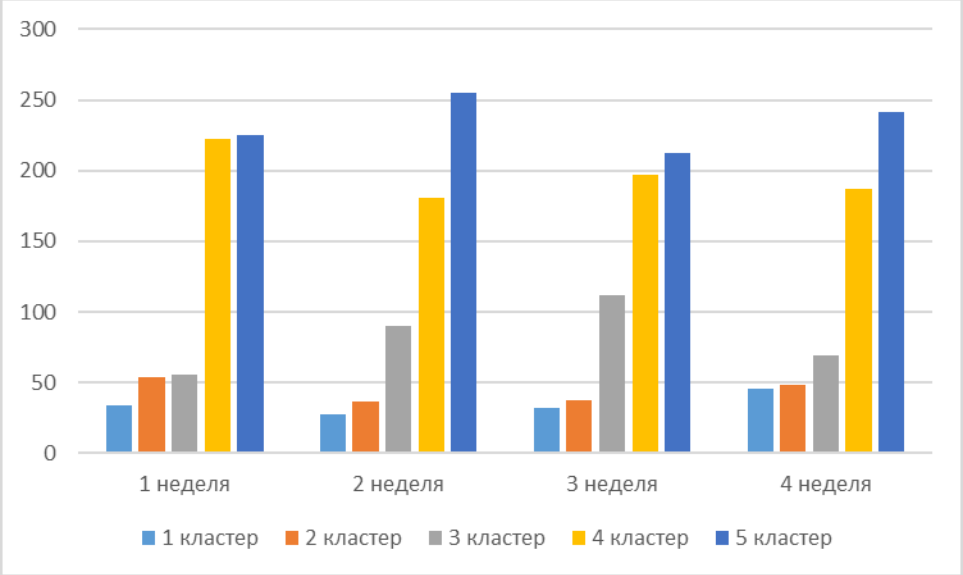

Рисунок 12 – Гистограмма разбиения пользователей по данным за 4 различные недели на 5 кластеров

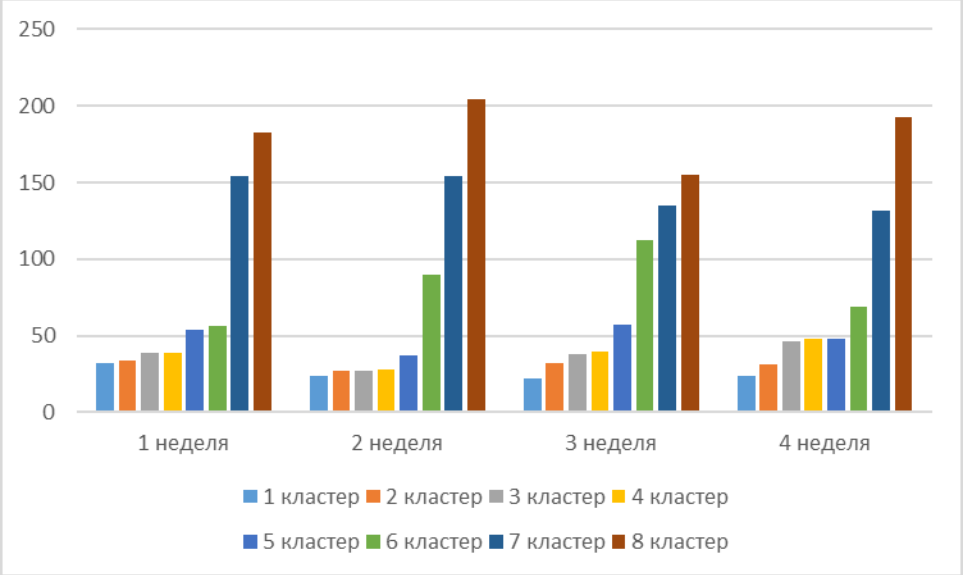

Рисунок 13- Гистограмма разбиения пользователей по данным за 4 различные недели на 8 кластеров

Для формирования устойчивых групп на множестве состоящем из m – пользователей, имея множество разбиений X_j состоящее из N разбиений за различные промежутки времени, каждое из которых разбито на k кластеров,

для каждого пользователя с номером $i \in [1;n]$, участвующего в кластерном анализе, можно составить такой вектор чисел $A_i = (a_1,\dots a_N)$, в котором каждая координата $0 \le a_j < k$ обозначает номер кластера, к которому принадлежит данный пользователь в разбиении X_j. Существование и единственность и такого вектора для каждого пользователя докатывается тем, что выбранный иерархический метод кластеризации принадлежит к типу четких методов, где каждому объекту выборки соответствует один и только кластер, к которому он принадлежит.

Сравнивая описанные выше вектора принадлежности кластерам для двух выбранных пользователей, можно судить о сходстве их предпочтений или их различии. При совпадающих векторах, или незначительно отличающихся, можно считать, что интересы пользователей схожи, а в противном случае считать их различными.

Максимальное количество групп пользователей, в том числе групп состоящих из одного человека, определяется по формуле $K = \max(m, k^N)$, где m, N и k – число пользователей, разбиений и кластеров соответственно.

При необходимости выделить группы пользователей, у которых вектора принадлежности идентичны, можно воспользоваться следующим алгоритмом (рис. 14):

1) Выбрать пользователя, не принадлежащего ни одной из групп

2) Для выбранного пользователя выбрать множество, состоящее из кластеров, к которым он принадлежит в каждом разбиении

3) Найти пересечение этих кластеров, которое в результате будет состоять из пользователей, имеющих одинаковые вектора принадлежности, и пометить это множество как отдельную группу

4) Если каждый пользователь имеет свою группу – завершить работу, иначе перейти к шагу 1.

Рисунок 14 - Алгоритм выделения устойчивых групп пользователей

В результате анализа различных разбиений было выявлено, что крупные группы пользователей присутствуют в любом разбиении, и достоверность таких групп напрямую зависит от количества разбиений и кластеров.

Таблица 4, отображает размер групп, которые были выявлены среди 591 пользователя прокси сервера.

Таблица 4 – характеристики группы (не единичных) для различных разбиений

Кол. разбиений	Кол. кластеров	Кол. групп	Средняя группа	Макс. группа
2	3	9	66	208
2	5	18	33	148
2	8	34	17	111
3	3	20	29	153
3	5	38	15	100
3	8	60	9	71

4	3	37	16	116
4	5	55	10	88
4	8	74	7	63
4	25	77	4	25

На рисунке 15 отображены расположение устойчивых групп, полученных из разбиений за 2 последующие недели на 3 кластера. Как видно из рисунка, больше половины пользователей из кластеров 1 и 2 не изменили свой кластер за 2 недели.

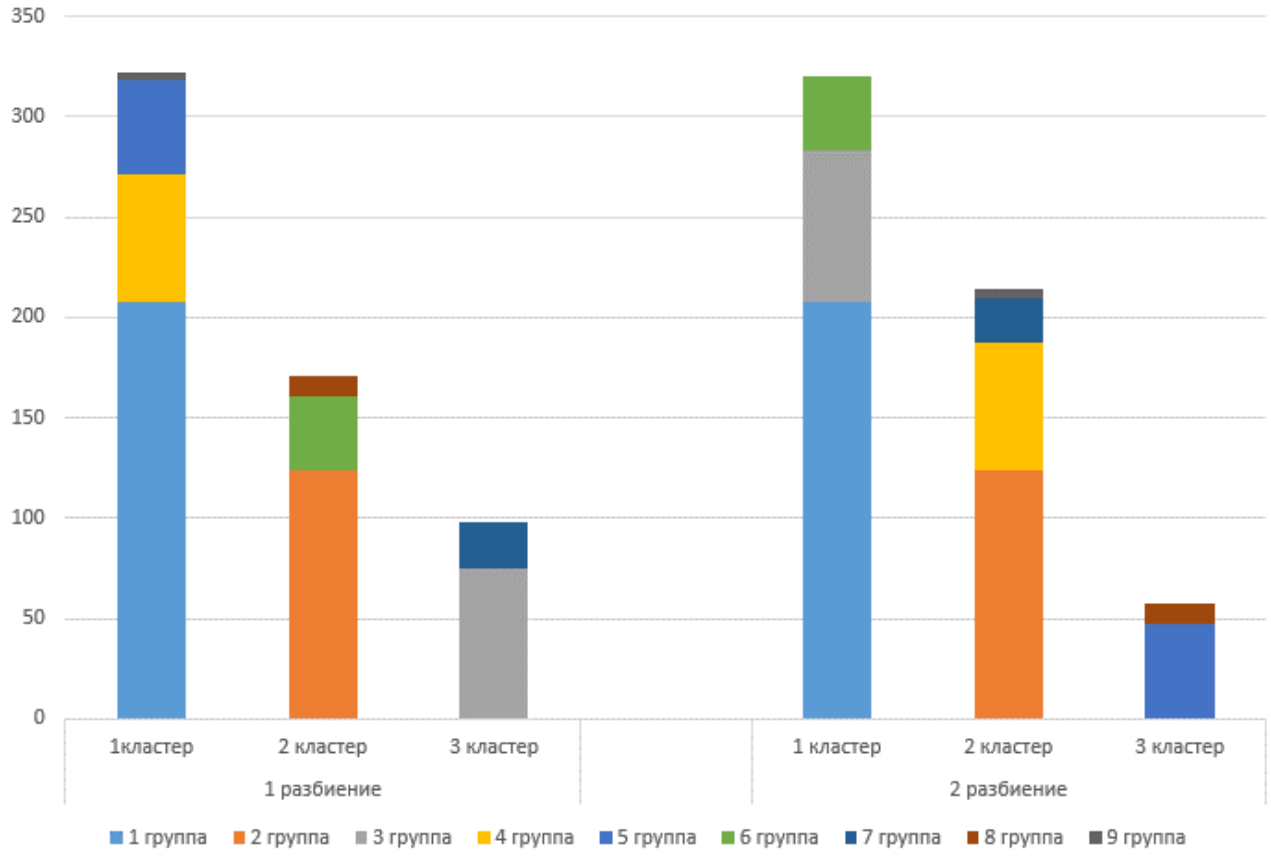

Рисунок 15 – Расположение устойчивых групп в различных разбиениях

Найденные группы можно использовать в различных целях: в сетевом-маркетинге, организации эффективной командной работы, в поиске отклонения поведения пользователя от нормального и для решения других интересных задач.

ЗАКЛЮЧЕНИЕ

В работе получены следующие результаты:

1. обработаны записи журнала прокси-сервера за период равный одному месяцу;
2. выполнена предобработка на основе Яндекс-каталога, позволившая снизить размерность пространства кластеризации 4000 до 160;
3. произведена иерархическая кластеризация пользователей прокси-сервера, по популярности тематик сайтов для каждого пользователя;
4. выявлены оптимальные значения числа кластеров;
5. выделены устойчивые группы пользователей со схожими предпочтениями.

Полученные результаты могут быть использованы для решения различных задач с использованием кластеризации слабоструктурированных данных.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Бериков, В. С. Современные тенденции в кластерном анализе / В. С. Бериков, Г. С. Лбов. – Всероссийский конкурсный отбор обзорно-аналитических статей по приоритетному направлению "Информационно-телекоммуникационные системы", 2008. – 26 с.
2. Ганенкова, Е. Г. Функциональный анализ: основные классы пространств / Е. Г. Ганенкова, К. Ф. Амозова. – Петрозаводск: ПетрГУ, 2013. – 26 с.
3. Ершов, К. С. Анализ и классификация алгоритмов кластеризации / К. С. Ершов, Т. Н. Романова. // Новые информационные технологии в автоматизированных системах. – 2016. – №19. – С. 274-279.
4. Котов, А. Кластеризация данных [Электронный ресурс]. – Режим доступа: http://logic.pdmi.ras.ru/~yura/internet/02ia-seminar-note.pdf.
5. Мандель, И. Д. Кластерный анализ / И. Д. Мандель. – М.: Финансы и статика, 1988. – 176 с.
6. Протасов, С.С. Как большие данные стали одной из самых интересных задач IT-индустрии [Электронный ресурс]. – Режим доступа: http://andrew--r.github.io/bigdata/.
7. Суслов, С. А. Кластерный анализ: сущность, преимущества и недостатки / С. А. Суслов. // Вестник НГИЭИ. – 2011. – №1. – С. 51-56.
8. Blanco-Silva, F. J. Learning SciPy for Numerical and Scientific Computing / F. J. Blanco-Silva. – Packt publishing, 2015. – 150 p.
9. Downey, A. B. Think Python: An Introduction to Software Design / A. B. Downey. – O'Reilly Media, 2002. – 300 p.
10. Duran, B. S. Cluster Analysis - A Survey / B. S. Duran, P. L. Odell. – Springer, 1974. – 146 p.
11. Jain, A. K. Data Clustering: A Review / A. K. Jain, M. N. Murty, P. J. Flynn. – ACM Computing Surveys, 1999. – 323 p.
12. Lutz, M. Programming Python / M. Lutz. – O'Reilly Media, 1996. – 1632 p.

13. McKinney, W. Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython / W. McKinney. – O'Reilly Media, 2012. – 429 p.
14. Müller, A. C. Introduction to Machine Learning with Python: A Guide for Data Scientists / A. C. Müller, S. Guido. – O'Reilly Media, 2016. – 394 p.
15. VanderPlas, J. Python Data Science Handbook: Essential Tools for Working with Data / J. VanderPlas. – O'Reilly Media, 2016. – 548 p.
16. Wessels, D. Squid: The Definitive Guide / D. Wessels. – O'Reilly Media, 2004. – 466 p.
17. Wessels, D. Internet Cache Protocol (ICP), version 2, 1997 [Электронный ресурс] / D. Wessels, K. Claffy – Режим доступа: https://tools.ietf.org/html/rfc2186.